TI hardware guide for programmer

History :

Date	Version	Modifications
05/03/09	1.0	initial release (pre-3)
05/05/05	1.1	Fixes and updates
14/07/05	1.2	Added RTC (HW3)

Contents

1 Introduction

I decided to write this guide while I was working on TiEmu. I needed a such documentation because there were a lot of small documentation spread around but no one which was complete and up-to-date.

This guide is a compilation of several documentation and may not be as complete as the following one :
- LowLevel.txt from FARGO package (TI92),
- io-ports.txt from Mxm's TiE (more up-to-date TI92 & TI92 v2),
- ti92.doc from Gareth James ("TI89 and TI92 structures, variables and information"),
- j89hw.txt and system.txt from Johan Eilert's ("TI89 documentation"),
- MC68000 user's manual and FLASH data sheet from Epson,
- Dan Englender's WiKi (TI84+ & Titanium, RTC & USB).

Maybe you should read them before…

Other information (from K. Kofler, O. Armand or L. Debroux) have been added to complete it.
Good doc can be found at http://tict.ticalc.org/documents.html, too.

This guide aims to be as exact as possible but may contains errors or mis-understanding. You are strongly encouraged to mail me if you have found an error or have more information...

If you have any suggestions or additions please email me at : roms@lievin.net.
Updates of this document will be available at : http://www.ticalc.org.

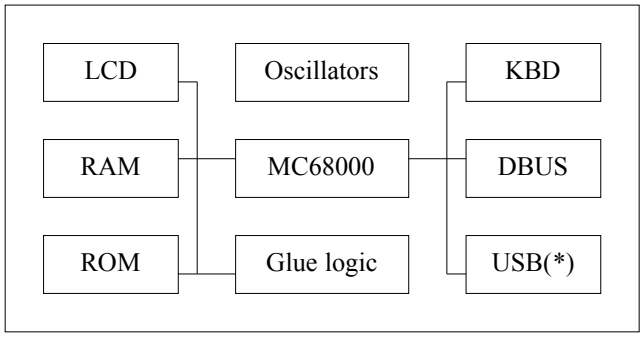
This guide has been written with OpenOffice 1.1.4 (with .DOC and .PDF export capability).

Enjoy !

2 Overview

The following guide covers the following calculators : TI89, TI92 (II), TI92+, V200 and Titanium.

The TI hand-helds covered by this documentation are m68k-based calculators only. They are built around a Motorola MC68000 processor and some peripherals with some glue-logic (ASIC). The picture below shows this:



(*) Titanium only.

Calculators can be classified into 3 categories: HW1, HW2 and HW3. Each category defines an hardware platform. Software is more or less common to all platforms although FLASH calculators (TI89, 92+, V200, Titanium in chronological order) introduced a gap with first generation of calculators like TI92 (II).
Note: unless explicitly specified, HW2 implies HW3.

	HW1	HW2	HW3
TI92	X		
TI92 II		X?	
TI92+	X	X	
TI89	X	X	
V200		X	
TI89 Titanium			X

2.1 Oscillators

HW1 calculators have two (HW2 has three) separate oscillators, they are referred to as OSC1 and OSC2 (and OSC3).

OSC1 is the M68000 CPU clock.
HW1: ~10 MHz
HW2: ~12 MHz

OSC2 is the timing base for the timers, the link I/O hardware and (HW1 only) the LCD controller.
HW1: ~680 kHz - ~770 kHz
HW2: ~520 kHz (= 2^19 Hz)

OSC3 (HW2 only) is the timing base for the LCD controller.
HW2: ~680 kHz

The speeds of OSC1 and OSC2 seem to be independent of the battery strength, but OSC3 runs slower with older batteries.

The speed of OSC2 can be calculated using this formula (assuming the APD time is the default) :
HW1: OSC2 speed = 242,688,000 / APD_seconds
HW2: OSC2 speed = 162,816,000 / APD_seconds
APD_seconds should be somewhere in the range 300-360 seconds. If the APD time is way longer than 400 seconds on a HW2, the HW1 formula should be used instead. This will happen if, for example, an ignorant game reprograms the HW2 timer with the HW1 settings.

2.2 Display

A black and white LCD display :
- 100 lines of 160 pixels for the TI-89,
- 128 lines of 240 pixels for the others.

2.3 Keyboard

The keyboard is accessed internally as a matrix.
The [ON] key is special, and is not part of the matrix.

2.4 RAM

Calculators usually have 256 KB of RAM (except for TI92).

Model	Memory (in KB)	Base address	End address	Mirrored	Repeated (times)
TI92	128	$000000	$1FFFFF	no	16
TI92 II	256	$000000	$3FFFFF	no	8
TI89	256	$000000	$3FFFFF	no	8
TI92+	256	$000000	$3FFFFF	no	8
V200	256	$000000	$3FFFFF	no	8
TI89 Titanium	256	$000000	$03FFFF	$200000 & $400000	no

Repeated means that the 256 KB RAM range is visible/accesible 8 times in the 2MB segment at the following addresses: $000000, $040000, $080000, $0C0000, $100000, $140000, $180000, $1C0000.

Mirrored (or ghosted) means that the RAM segment is accessible from other segment ($200000 for instance).

Note: The TI-92 is capable of addressing 256 kb of interleaved RAM. In this case, it would be addressed in the range 000000-03FFFF. The European language upgrade module (ROM 2.1 / TI-92 II) and the TI-92 Plus Module take advantage of this capability, as well as expanding the ROM to 2 MB (400000-5FFFFF).

2.5 ROM

TI92 (II) has EPROM, others have FLASH memory. Memory size is :

Model	Memory (in MB)	Base address	End address	Mirrored
TI92	1	$200000 or $400000	$2FFFFF or 4FFFFF	no
TI92 II	2	$400000	$5FFFFF	no
TI89	2	$200000	$3FFFFF	no
TI92+	2	$400000	$5FFFFF	yes (hw2)
V200	4	$200000	$5FFFFF	no
TI89 Titanium	4	$800000	$AFFFFF	no

TI92 ROM can be mapped at 2 different base addresses depending on the manufactured model.

TI92+ HW2 has its ROM mirrored on $200000, too.

2.6 Link-port

The link-port uses a proprietary interface called D-BUS. See this complete documentation : TI Link Guide.

2.7 Timers

Every model has two (HW1/3) or three (HW2) clocked sources. One (two) of them work at a fixed rate, the last one is programmable:
- AI1 is triggered at OSC2 / 2^{11},
- AI3 is triggered at OSC2 / 2^{19} (not available on HW1/3, used by USB),
- AI5 is triggered at a variable rate (prescaler + timer).

Every model has a programmable prescaler (except for TI92 & TI92 II) with a programmable counter/timer which is incremented on an OSC2 basis. Prescaler ratios are : 2^5, 2^9, 2^12 and 2^18.

2.8 USB

A great work is being done by Olivier Armand (Titanium) and Dan Englender (TI84+).
This section will be updated later but you can take a look at the Dan's WiKi.

2.9 RTC

A new device has been added on HW3 : a Real Time Clock. At the time being, there are few informations available on the Dan's WiKi…

2.10 Glue logic

The glue-logic is responsible of the peripherals and their interconnection with the CPU. At the time being, all of this is placed in an ASIC. The TI92 model is the only model with a true/real M68000, others have a custom chip.

All of the glue-logic is accessible from memory space:

	HW1/2/3		HW2/3	
Model	Size	Base address	Size	Base address
TI92	32	$600000		
TI92 II	32	$600000		
TI89	32	$600000	32	$700000
TI92+	32	$600000	32	$700000
V200	32	$600000	32	$700000
TI89 Titanium	32	$600000	256 256	$700000 $710000

Note: all $600000 ports are repeated but $700000 aren't.

Registers: all others.

3 Memory mapping

Memory is mapped as follows:

Address	TI92	TI92 II	TI92+	TI89	V200	Titanium

000000-1FFFFF	RAM					
200000-3FFFFF	Int. ROM		ghost ROM	ROM	ROM	ghost of RAM
400000-5FFFFF	Ext. ROM	Ext. ROM	ROM			
600000-7FFFFF	I/O	I/O	I/O	I/O	I/O	I/O
800000-9FFFFF						ROM
A00000-BFFFFF						
C00000-DFFFFF						
E00000-FFFFFF						

But, memory is used as follows (begin-end or begin (size)):

Contents	TI92	TI92 II	TI92+	TI89	V200	Titanium
RAM	$000000 $01FFFF	$000000-$03FFFF (256KB)				
ROM (internal)	$200000 $2FFFFF	$200000 $3FFFFF	ghost (HW2)	$200000 $3FFFFF	$200000 $5FFFFF	$800000 $AFFFFF
ROM (external)	$400000 $4FFFFF	$400000 $5FFFFF	$400000 $5FFFFF			
ASIC	$600000 (32)	$600000 (32)	$600000 (32)	$600000 (32)	$600000 (32)	$600000 (32)
ASIC (HW2)			$700000 (32)	$700000 (32)	$700000 (32)	$700000 (256)
ASIC (HW3)						$700000 (256)

Some TI92 model have ‘internal’ memory, some others have ‘external’ but they can’t have both. This depends on the manufacturing.

Note: the 68000 processor keeps a 256(d) byte exception vector table at the bottom of RAM at $000000.

Note 2 : the TI-92 is capable of addressing 256 KB of interleaved RAM. In this case, it would be addressed in the range 000000-03FFFF. The European language upgrade module (ROM 2.1 / TI-92 II) and the TI-92 Plus Module take advantage of this capability, as well as expanding the ROM to 2 MB (400000-5FFFFF).

4 Interrupts

AutoInt	Usage
1	Triggered at a fixed rate: OSC2/2^11. It runs at approximately 350 Hz on HW1. See $600015.
2	Triggered when the *first* unmasked key (see $600019) is *pressed*. Keeping the key pressed, or pressing another without released the first key, will not generate additional interrupts.
3	TI92: print "Level 3 Autovector" and freeze. Never triggered. HW1/2: disabled by default by AMS, early versions crash if it is enabled. When enabled, it is triggered at a fixed rate: OSC2/2^19. See $600015:2. HW3: used by USB port.
4	Triggered by the link hardware for various reasons.

5	Triggered by the programmable timer when $600017 is set to its initial value. The default rate (set by AMS on reset) is OSC2/(K*2^9), where K=79 for HW1 and K=53 for HW2. After ROM initialization, it runs at approximately 18 Hz. See $600015 and $600017.
6	Triggered when [ON] is pressed.
7	If the "vector table write protection" is enabled, this interrupt is triggered on a write access to any address below $000120. See $600001:2.

For auto-interrupts 1, 3 and 5, see §6.6.

5 View per I/O ports

This part is the most one. It contains all the information needed to access registers and settings. This part focus on registers. The part afterwards focus on devices.

Legend:

Model (access) : default value

Bit	Usage
X	Some text.

Access:
R = data can be read at any time,
W = data can be written at any time.

Reminder:
- ports at $600000 are HW1/2/3,
- ports at $700000 are HW2/3,
- ports at $710000 are HW3 only.

5.1 $600000 : contrast and battery status

TI92 (RW)

Bit	Usage
5	Bit #0 of contrast

TI89 (RW) : $00

Bit	Usage
7	Keep to 0
6	Keep to 0
2	If set, battery voltage level is *above* the trig level (see $600018).

5.2 $600001 : stack overflow detection

TI92 (RW)

Bit	Usage
2	If set, generate Auto-Int 7 when memory below [000120].
0	If clear: interleave RAM (allows use of 256K of RAM).

TI89 (RW) : $04

Bit	Usage
2	Write protect vector table ($000000-$00011F). Int 7 will also be generated on writes to this address range, and on writes to $E00000-$FFFFFF.
0	(HW1) Bit cleared means 256K RAM, bit set means 128K RAM. Consider this bit "read-only" and keep it clear, or else the RAM will not function properly!

5.3 $600003 : wait-states generator

TI89 (WO) : $FF

Bit	Usage
	Bus waitstates. The TI89 hardware needs no waitstates. AMS messes with this port on startup for compatibility with the TI92, but the battery checker will reset it to $FF within one second.
6-4	Wait states =(7-n) for non-RAM accesses.
2-0	Wait states =(7-n) for RAM accesses.

5.4 $600005 : idle/wake-up control

TI92 (WO)

Bit	Usage
	Turn off OSC1 (and the CPU), wake on int level 6 (ON key) and ...
4	int level 5 (programmable timer)
3	int level 4 (link)
2	
1	int level 2 (keyboard)
0	int level 1 (OSC2/2^11)

TI89 (WO)

Bit	Usage
	Turn off OSC1 (and the CPU), wake on int level 6 (ON key) and ...
4	int level 5 (programmable timer)
3	int level 4 (link)
2	int level 3 (OSC2/2^19)
1	int level 2 (keyboard)
0	int level 1 (OSC2/2^11)

5.5 $60000C : link control

TI92 (RW) : $8D

Bit	Usage
7	Autostart enable.
6	Link disable.
5	Link timeout disable.
4	
3	Autostart interrupt enable.
2	Autostart interrupt enable.
1	TX buffer empty interrupt enable.
0	RX buffer full interrupt enable.

See this complete documentation : TI Link Guide.

5.6 $60000D : link status

TI92 (RW)

Bit	Usage
7	Link error.
6	TX buffer empty.
5	RX buffer full.
4	Link interrupt.
3	Autostart.
2	
1	
0	

See this complete documentation : TI Link Guide.

5.7 $60000E : low level link access

TI92 (RW)

Bit	Usage
7	
6	
5	
4	
3	D1 in
2	D0 in
1	D1 out
0	D0 out

See this complete documentation : TI Link Guide.

5.8 $60000F : link rx/tx register

TI92 (RW)

Bit	Usage
7-0	RX/TX buffer

See this complete documentation : TI Link Guide.

5.9 $600010/11 : LCD address

TI92 (WO)

Bit	Usage
15-0	Address of LCD memory divided by 8.

TI89 (WO) : $0980

Bit	Usage
15-0	HW1: LCD frame buffer address, divided by 8. This address is latched into the LCD controller DMA address register on each FS (i.e. at the beginning of every frame). HW2: No function. See $700017.

5.10 $600012 : LCD width

TI92 (WO) : $31

Bit	Usage
7-0	LCD horizontal frequency ?

TI89 (WO)

Bit	Usage
5-0	LCD logical width =(64-n)*2 bytes =(64-n)*16 pixels. The LCD controller DMA will send this many pixels to the screen for each line (= between each RS).

5.11 $600013 : LCD height

TI92 (WO)

Bit	Usage
7-0	$100 – number of LCD scanlines. If the number of scanlines is smaller than $80 (the actual height of the LCD) the display will be duplicated in the lower half. Decreasing the height darkens the LCD; increasing the height lightens the LCD. The TI-92's ROM writes $80 to this port during initialization.

TI89 (WO)

Bit	Usage
7-0	LCD logical height =(256-n). Number of "row syncs" (RS) between each "frame sync" (FS). (This is the (logical) screen height).

5.12 $600015 : timer & OSC2/3 control

TI89 (RW) : $1B

Bit	Usage
7	Master disable timer interrupts (level 1, 3 and 5)
5-4	Increment rate of $600017 %00: OSC2/2^5 %01: OSC2/2^9 (AMS default) %10: OSC2/2^12 %11: OSC2/2^18
3	Enable incrementing of $600017
2	Trigger interrupt level 3 at OSC2/2^19 (~1 Hz on HW2)
1	OSC2 (and OSC3?) enable (bit clear means oscillator stopped!).
0	LCD controller DMA enable, LCD blank ("white") if 0. This bit is only examined by the hardware at the start of each frame. HW1: The DMA steals ~10% of the CPU bus bandwidth.

5.13 $600017 : programmable timer

TI92 (RW) : $B2

Bit	Usage
7-0	Programmable rate generator. Set the timer's initial value by writing it to this port. The timer is incremented every 6250 clock cycles, unless it has a value of zero, in which case it is reset to its initial value. The LCD is refreshed every 16th time this timer is incremented.

TI89 (RW) : $B2 (hw1) or $CC (hw2)

Bit	Usage
7-0	READ: Read the current value. WRITE: Set the initial (and current) value for the timer. The timer value is incremented at the rate specified at $600015 and triggers interrupt level 5 when it "overflows" to $00. The next increment forces the timer to reload the initial value. The count sequence looks like this: value, value+1, ..., $FF, $00 (interrupt!), value, value+1, ... To trigger an interrupt every 'n'th increment, write '257-n' to this register.

5.14 $600018 : keyboard row mask

TI92 (RW)

Bit	Usage
1-0	Keyboard row mask; setting a bit masks the corresponding row of the keyboard from being read by [60001B].

TI89 (RW) : $80

Bit	Usage
1-0	Battery voltage trigger level.

5.15 $600019 : keyboard row mask

TI92 (RW)

Bit	Usage
7-0	Same as above.

TI89 (RW)

Bit	Usage
6-0	Keyboard row mask, bit =1 masks key row (vertical) from being read at $60001B and generates interrupt on key pressed.

5.16 $60001A : ON key status

TI92 (RO)

Bit	Usage
1	ON key status (0=down, 1=up).

TI89 (RW)

Bit	Usage
1	READ: current status of the [ON] key, =0 if pressed.
7-0	WRITE: acknowledge [ON] key interrupt (level 6).

5.17 $60001B : keyboard column state

TI92 (RO)

Bit	Usage
7-0	Keyboard column mask; if a bit is clear, one or more keys in the corresponding column are being held down. Keys in rows masked by [600018] are ignored.

TI89 (RW)

Bit	Usage
7-0	READ: Keyboard column input, each bit =1 means ALL keys in the corresponding key column are UP (not pressed). See $600019. WRITE: Acknowledge keyboard interrupt (level 2).

5.18 $60001C : LCD RS frequency

TI92 (WO) : $21

Bit	Usage

7-0	

TI89 (WO) : $21

Bit	Usage
5-2	LCD RS (row sync) frequency, OSC2/((16-n)*8). %1111 turns off the RS completely (used when LCD is off).

5.19 $60001D : contrast

TI92 (WO)

Bit	Usage
3-0	Bits 4321 of contrast.

TI89 (WO) : $80

Bit	Usage
7	HW1: Voltage multiplier enable. Keep set (=1).
4	HW1: Screen disable (power down). HW2: LCD contrast bit 4 (MSb).
3-0	LCD contrast bits 3-0 (bit 3 is MSb on HW1).

Note :
- TI92, V200 : contrast decreases => LCD is darker,
- TI89, Titanium : contrast increases => LCD is lighter.

5.20 $700000 : RAM page execution protection bitmask

$FFDF FFFF FFFF FFFF = allow execution at $005xxx only. Bit SET means instruction fetches NOT allowed in that 4K page. The Protection must be disabled for changes to have any effect.

TI89 (RW)

Bit	Usage
15-0	$00Fxxx-$000xxx (one bit for each 4K page).

5.21 $700002 : RAM page execution protection bitmask

TI89 (RW)

Bit	Usage
15-0	$01Fxxx-$010xxx (one bit for each 4K page).

5.22 $700004 : RAM page execution protection bitmask

TI89 (RW)

Bit	Usage
15-0	$02Fxxx-$020xxx (one bit for each 4K page).

5.23 $700006 : RAM page execution protection bitmask

TI89 (RW)

Bit	Usage
15-0	$03Fxxx-$030xxx (bit 15 controls $1FFFFF-$040000 too).

5.24 $700008 to 70000F : RAM page execution protection bitmask

These ports are the ghosts of the ports for the protection of execution in RAM ($700000 to 700007). Any writing to either the real ports or the ghosts will change both ports (eg $700002 and $70000A).

5.25 $700012 : FLASH execution protection

TI89 (RW)

Bit	Usage
5-0	First exec protected flash ROM sector =(n*$10000+$210000). The Protection must be disabled for changes to have any effect.

5.26 $700014 : Real Time Clock (HW2)

TI89 (RW)

Bit	Usage
15-0	Incremented every 2^13 = 8192 seconds exactly. The whole word must be read : reading the port byte by byte can return wrong values. The timer is not incremented when the batteries have been removed, but the value it had when they were removed is kept. Removing the lithium battery and putting it back gives a random value to the timer.

5.27 $700017 : LCD address/range

TI89 (RW)

Bit	Usage
1-0	The HW2 display controller doesn't fetch pixel data from RAM but from its own memory. This memory is 4K and all CPU writes to the selected address range (below) will go to both RAM and this memory. (NOTE: It is not possible to *read* from the display memory.) %00: $4C00-$5BFF %01: $5C00-$6BFF %10: $6C00-$7BFF %11: $7C00-$8BFF Note: AMS never initializes this register, it simply assumes that the display controller is snooping writes to $4C00-$5BFF. Note: For obvious reasons, the contents of the screen does not change when you write to this register. This register is NOT the HW2 equivalence of the register at $600010 on HW1.

5.28 $70001D : LCD control/status

TI89 (RW)

Bit	Usage
7	Toggles every FS (every time the LCD restarts at line 0)
3	Battery checker bit B
1	Screen enable (clear this bit to shut down LCD)
0	Battery checker bit A

5.29 $70001F : RTC control (HW2)

TI89 (RW)

Bit	Usage
2-1	enable the incrementing of $700014.w when bit 1 is set. Set by AMS on reset. bit2/bit1 1 1 : State set by AMS on reset. $700014.w is incremented. 1 0 : Auto-ints 1, 2, 3 and 5 are inhibited. The RTC is stopped. 0 1 : The RTC is stopped. The auto-ints works normally. 0 0 : The RTC is stopped. The frequencies of all the auto-interrupts are lower (OSC2 must be slower): AI1 : ~ 176 Hz instead of 256 Hz. AI3 : ~ 40 ticks per minute instead of 60. AI5 : ~ 13.2 Hz instead 19.3 Hz.
0	Use 5 contrast bits (default for AMS). When this bit is cleared, the contrast hardware uses $60001D:3-0 only to specify the contrast, effectively emulating a HW1. AND, clearing this bit activate the execution in areas from which the Protection can be disabled : ROM_BASE+$12000 to ROM_BASE+$17FFF and ROM_BASE+$1A000 to ROM_BASE+$1FFFF. The execution in the boot sector is not protected by the protection

5.30 $710040 : Real Time Clock (HW3)

None of these ports are protected by the Flash/IO protection.

TI89t (WO)

Bit	Usage
31-0	Value of RTC (seconds since January 1st, 1997 00:00:00).

5.31 $710044 : RTC unknown

TI89t (WO)

Bit	Usage
7-0	???

5.32 $710046 : Real Time Clock (HW3)

TI89t (RO)

Bit	Usage
31-0	Value of RTC (seconds since January 1st, 1997 00:00:00).

5.33 $71005f : RTC Control

TI89t (RW)

Bit	Usage
1	Set to 1 in normal operation. AMS toggles this bit to 0 and back to 1 to set the clock (falling edge). This operation copies: $710040.l to $710046.l.
0	Enable (1) or disable (0) the RTC.

6 View per devices

This part takes over the previous one but is focuses on devices. This is a ‘data sheet view’ rather than a ‘reverse-engineered view’.

6.1 OSCx

OSC1 can be turned off by $600005 and restarted on interrupt 1..6.

OSC2 (and OSC3) can be turned off by $600015:1.

6.2 LCD

6.2.1 HW1

The HW1 display system is divided into two parts: the controller and the LCD. The timing source is OSC2, so it is pretty easy to synchronize the screen update with an interrupt (1 or 5) for flicker-free grayscale.

It takes one OSC2 cycle to transfer four bits (pixels) to the screen, hence the DMA reads one byte from RAM every other OSC2 cycle. The default width of 240 pixels (see $600012) thus takes 60 OSC2 cycles, leaving 4 idle cycles at the end of each pixel row (see $60001C). At the start of each row a "row synchronization" signal (called RS) is sent to the LCD. This is needed because only the first 160 pixels will be displayed on the LCD and it must somehow know when these 160 pixels start. Note that the RS generation and the logical width are completely independent of each other: It is possible to set the RS rate too fast compared to the time needed to transfer all the pixels for a line! The result is very unpredictable. In fact, the RS will only be honored by the DMA when it is idle -- but the LCD always sees it and acts upon it! For the special case when the DMA is finished at the very same time as the RS is generated, the DMA will skip the last byte (it will not be sent to the LCD) and (properly) start sending the next screen line instead.

When 128 pixel rows (see $600013) have been transferred in this manner, everything starts over and the "frame synchronization" signal (called FS) is activated to tell the LCD to start over from the top again. At the same time, the DMA restart at the top of the frame buffer (see $600010). (If the DMA is ignoring some of the RS because of improper programming (as mentioned above), it will ignore all FS that occur at the same time as these RS as well!)

When there are less than 100 lines between each FS, the displayed image will be "repeated." E.g. if the logical screen height is set to 45 lines, the top of the image will be seen on row 0, row 45 and row 90. (The reason for the duplication is that the row driver is in fact a 100-bit shift register where FS is shifted in at the top every RS.)

The frame rate can be calculated:
  logical_screen_height * RS_interval = /with default settings:/ = 128*64 =
   = 8192 OSC2 cycles/frame (= ~90 Hz)
Note that it is only the logical screen height and the contrast that affect the overall "darkness" of the screen.

The highest possible (full-screen) frame rate is 4800 OSC2 cycles per frame.

6.2.2 HW2

The HW2 display controller works very much like its HW1 counterpart, but with two major exceptions:

* The timing base is OSC3 which is *completely* unrelated to the timer and the interrupt system. Luckily, FS is available at $70001D:7 so synchronization (and thus flicker-free grayscale graphics) is possible.

* The display controller does not fetch pixel data from RAM, instead it has an internal 4K display buffer. The display controller monitors the bus activity and it updates the display buffer on every write to the snoop range (see $700017). In other words, writes to this range go to both RAM and the display buffer.

Registers: $600010, $600012.

6.3 Keyboard

As was hinted in the I/O ports section, the keyboard is accessed internally as a matrix. This matrix can be read by writing [600018], pausing to allow the I/O to recover, then reading [60001B].

Keyboard Matrix on the TI 92:

Row V Col>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit 0	down	right	up	left	hand	shift	diamond	2nd
Bit 1	3	2	1	F8	W	S	Z	unused
Bit 2	6	5	4	F3	E	D	X	unused
Bit 3	9	8	7	F7	R	F	C	STO
Bit 4	,	)	(	F2	T	G	V	space
Bit 5	TAN	COS	SIN	F6	Y	H	B	/
Bit 6	P	ENTER2	LN	F1	U	J	N	^
Bit 7	*	APPS	CLEAR	F5	I	K	M	=
Bit 8	unused	ESC	MODE	+	O	L	theta	backspc
Bit 9	(-)	.	0	F4	Q	A	ENTER1	-

Note: ENTER1 is on the alphabetic _and_ numeric keypads.
      ENTER2 is next to the cursor pad.

Keyboard Matrix on the TI 89:

Row V Col>	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Bit 0	Alpha	Diam.	Shift	2nd	Right	Down	Left	Up
Bit 1	F5	Clear	^	/	*	-	+	Enter
Bit 2	F4	Backspc	T	,	9	6	3	(-)
Bit 3	F3	Catalog	Z	)	8	5	2	.

```
| Bit 4 |   F2  |  Mode |   Y   |   (   |   7   |   4   |   1   |   0   |
| Bit 5 |   F1  |  Home |   X   |   =   |       |   EE  |  STO  |  Apps |
| Bit 6 |       |       |       |       |       |       |       |  Esc  |
+-------+-------+-------+-------+-------+-------+-------+-------+-------+
```

Because of the way the TI89/TI92+ 's keyboard is wired, if you hold down three keys that form the corners of a rectangle, the TI89/TI92+ will think you are also holding down the key at the fourth corner.

The [ON] key is special, and is not part of the matrix.

Registers: $600018, $60001A, $60001B.

6.4 RAM

The TI-92 is capable of addressing 256 KB of interleaved RAM : see $600001.

6.5 ROM

The FLASH ROM is an Epson device (LH28F160S3T or LH28F320 according to the amount of memory).

Writing to Flash EEPROMs is much slower than writing to RAM, and requires writing control codes to the address space of the chip to perform these operations. Apparently the Flash EEPROM used is the "SHARP LH28F160S3-L", which has many features (some may be disabled) including writing (multiple) words/bytes, erasing 64kb blocks, and locking of blocks.

[For more information on this chip see Johan Borg's file at http://d188.ryd.student.liu.se/ftp/calculator/ti89/tech/flashrom.txt, he also has put up the datasheet at ftp://d188.ryd.student.liu.se/calculator/ti89/tech/f160s3.pdf which is worth a look.]

Fortunately only a subset of this chips features are used; writing words and erasing blocks. Write operations take 12.95 micro-seconds, read operations 100ns, so you can see why a special mode of operation is required.

To program the EEPROM we need to enter a special mode by writing certain control codes to the address range. Two operations can be perfored, erase (64kb) sector, and write a word. In this write mode reading from the address range gives the status register of the EEPROM, generally the only value that is used is that bit 7 of the read word will be set when complete. After the operations are complete Read mode is then set by writing another control code.

The drawback of this method is that the ROM cannot be read whilst write operations are being performed. Therefore the writing code must execute from the RAM. Trap 11 handles all of this on the calculator, copying the relevent portions to a buffer then executing them.

The following subset of operations is used:

Code	Function
0x1010	Write Setup (next word will be written)
0x2020	Erase Setup
0x5050	Clear Status Register
0x9090	Read ID codes
0xD0D0	Erase Conform (address)

0xFFFF	Read Memory or Reset

Erasing and writing require to writes, e.g. 0x2020, 0xD0D0 to perform erase to prevent accidental operations.

Examples

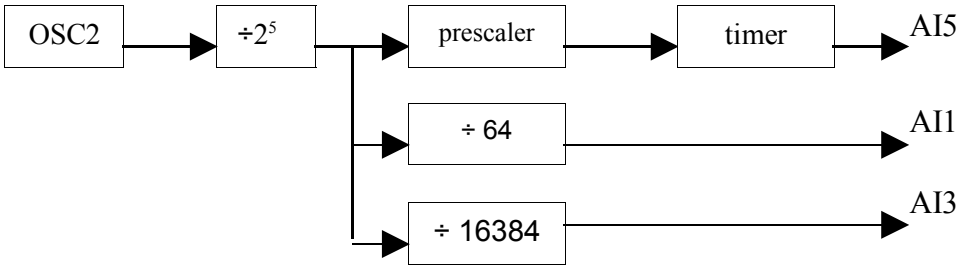
```
;erase 64kb block in which ERASE_ADDR resides
          lea       (ERASE_ADDR),a2
          move.w    #0x5050,(a2)        ;Clear Status Register
          move.w    #0x2020,(a2)        ;Erase Setup
          move.w    #0xD0D0,(a2)        ;Erase Confirm
write_state_busy:
          move.w    (a2),d0             ;Read Status Register
          btst      #7,d0               ;1=Ready
          beq       write_state_busy
          move.w    #0xFFFF,(a2)        ;Read
```

```
;write VALUE to WRITE_ADDR
          lea       (WRITE_ADDR),a2
          move.w    #0x5050,(a2)        ;Clear Status Register
          move.w    #0x1010,(a2)        ;Write Setup
          move.w    #VALUE,(a2)         ;Erase Confirm
write_state_busy:
          move.w    (a2),d0             ;Read Status Register
          btst      #7,d0               ;1=Ready
          beq       write_state_busy
          move.w    #0xFFFF,(a2)        ;Read
```

See the Epson LH28F160S3T (2MB) or LH28F320BF (4MB) data sheet for more information.

6.6 Timer

The TI calculator can provides you several ‘timers’. This is summed-up below:



Prescaler values : :÷1, ÷16, ÷128, ÷8192.

Useful bits/registers:
- master disable interrupts (1,3,5) by $60015:7,
- prescaler ratio: $600015:5-4,
- timer enabled by $600015:3,
- AI3 enabled by $600015:2,
- OS2/OSC3 can be turned off by $600015:1.

Timer can be read/write at $600017.

6.7 RTC

This device exists on HW3 only. The RTC keeps count of seconds since January 1st, 1997 00:00:00 with leap years (2000, 2004 => Feb the 29th).

7 Stealth I/O

7.1 Overview

The status of the various protections can only be changed when a global "lock" is deactivated. For historical reasons, this "lock" is called "the Protection" in this document (note the capital P).

Note: “consecutive accesses” means consecutive in *time*, not (necessarily) consecutive *addresses*.)
Note2: information is given for TI89 which have $200000 as FLASH ROM base address. Nothing is changed for other models apart base address.

The following protection features are *always* active, they cannot be disabled.

All HW models:
- write accesses to the boot installer sector ($200000-$20FFFF) are filtered and never reach the flash ROM. This sector is **also** permanently write protected by a feature of the flash ROM chip.

HW1 specific:
- any four consecutive accesses to $1C0000-$1FFFFF crashes the calculator,
- any three consecutive access to a flash ROM sector (64K) that is within the "multiple access forbidden" range crashes the calculator. Effectively inhibits the CPU from fetching instructions from the archive memory.

HW2/3 specific:
- an instruction fetch from a flash ROM sector (64K) that is within the "execution forbidden" range crashes the calculator. (See $700012).
- an instruction fetch from a RAM 4K page that has its "execution forbidden" bit set crashes the calculator. The protection hardware also considers the "shadow RAM" range $040000-$1FFFFF to be part of the last RAM page. See $700000.

The following protections are enabled only when the Protection is enabled.

All HW models:
- write accesses to the flash ROM ($210000-$3FFFFF) are filtered and never reach the flash ROM.
- the certificate memory ($210000-$211FFF) is read protected when the Protection is enabled. Read/write otherwise.
- the memory at $218000-$219FFF is read protected. Same as above.

HW2/3 specific:
- certain memory mapped I/O ports are locked and protected against modification. (These include, of course, the RAM page protection bit mask and the flash ROM executable sector limit.)

The following protections can be enabled and disabled at any time.

All HW models:
- write accesses to $000000-$00011F are filtered and trigger interrupt 7. See $600001.

7.2 Address space (again)

Detailled reminder on TI89:

Address	Size	Description
$000000-$1FFFFF	2 MB	RAM (256KB, repeated 8 times)
$000000-$00011F	288	Write protected
$200000-$3FFFFF	2 MB	FLASH ROM (write protected)
$200000-$20FFFF	64 KB	Boot code, always write protected
$210000-$211FFF	8 KB	Certificate memory (read protected)
$212000-$217FFF	24 KB	available for code/data
$218000-$219FFF	8 KB	(read protected ?!?)
$21A000-$3FFFFF	1944 KB	available for code/data

Stealth I/O on TI89:

Address	Size	Type	Use
$040000-$07FFFF	256 KB	stealth I/O (HW1 only)	Archive memory limit bitmask
$080000-$0BFFFF	256 KB	stealth I/O (HW1 only)	
$0C0000-$0FFFFF	256 KB	stealth I/O (HW1 only)	
$180000-$1BFFFF	256 KB	stealth I/O	Screen ON/OFF
$1C0000-$1FFFFF	256 KB	stealth I/O	Protection enable/disable
$200000-$20FFFF	64 KB	stealth I/O	Protection access authorization
$212000-$217FFF	64 KB	stealth I/O	
$21A000-$21FFFF	64 KB	stealth I/O	

7.3 Use

Apart from the dedicated (memory mapped) I/O, there are special "stealth" I/O ports that occupy parts of the RAM and the flash ROM address ranges.
Every access (read or write) to these special address ranges will issue a transparent stealth I/O operation as well as performing the usual access to RAM or flash ROM. To make things less complicated, it is only the *address* and the *type* of the access (HW1: READ or WRITE, HW2/3: READ/WRITE and CPU function codes FC2-FC0) that matters to a stealth I/O port, not the actual data. Be careful when WRITING to a stealth I/O port that occupies the same address range as RAM !

"N consecutive accesses [to a memory range]" means that there must be no other bus activity during these N accesses (they must be consecutive in *time*). Remember that the HW1 LCD controller DMA is constantly fetching pixel data from RAM. Use $600015:1 to disable it.

The Protection must be disabled for changes to have any effect.

$040000-$07FFFF: (HW1 only)
READ: clears archive memory limit bit 0
WRITE: sets archive memory limit bit 0
$080000-$0BFFFF: (HW1 only)
READ: clears archive memory limit bit 1
WRITE: sets archive memory limit bit 1
$0C0000-$0FFFFF: (HW1 only)
READ: clears archive memory limit bit 2
WRITE: sets archive memory limit bit 2

Three consecutive access to any address >=$390000+'limit'*$10000 and <$400000, where 'limit' is

the combined value of the three bits above, crashes the calculator. Use 'limit'=%111 to disable this protection altogether.

(HW2/3:? The CPU must be in supervisor mode for changes to have any effect.)

$180000-$1BFFFF: Screen power control
READ: Screen power off
WRITE: Screen power on
Note: The screen is automatically powered up on reads from the range

$200000-$3FFFFF. (However, $60001D:4 is always obeyed.)

$1C0000-$1FFFFF: "the Protection" enable/disable
Note: Four consecutive accesses to this range crashes a HW1 calc!
READ: Enable the Protection
WRITE: Disable the Protection
Note: No access to this range will have any effect unless the access is "authorized," see below.

$200000-$20FFFF,
$212000-$217FFF,
$21A000-$21FFFF: "the Protection" access authorization
HW1:
In order to alter the state of the Protection, THREE consecutive read accesses must occur from any of the three ranges above immediately prior to the access to the Protection enable/disable range.
HW2/3:
(Note: This is somewhat complicated and I don't know exactly how it works. The procedure listed below is probably too strict. Anyway, this is what AMS and the boot sector does.)
In order to alter the state of the Protection, (at least) SEVEN supervisor instruction fetches must occur from any of the three ranges above prior to the access to the Protection enable/disable range. The choice of instructions is not arbitrary, it must be one of the following sequences:
* To DISABLE the Protection: $4E71, <multiple non-instruction accesses are allowed here>, $4E71, $4E71, $46FC, $2700, $3080, <any instruction here>. The very next access must be the WRITE to the Protection enable/disable range.
* To ENABLE the Protection: $4E71, <multiple non-instruction accesses are allowed here>, $4E71, $4E71, $46FC, $2700, $3010, <any instruction here>. The very next access must be the READ from the Protection enable/disable range.
($4E71="nop", $46FC="move #imm16,sr", $3080="move.w d0,(a0)", $3010="move.w (a0),d0")

End.